Wireless Local Area Network Security: A Framework for Repairing the Broken WEP Protocol

Russ Housley
Jesse Walker
Nancy Cam-Winget

1. Introduction

The IEEE 802.11 [6] standard introduced the Wired Equivalent Privacy (WEP) algorithm to protect communication from eavesdropping and to prevent unauthorized wireless network access. WEP has critical security flaws, as recently published by Nikita Borisov, Ian Goldberg, and David Wagner [1] and Scott Fluhrer, Itsik Mantin, and Adi Shamir [2]. The flaws are the result of incorrectly using the RC4 stream cipher and poorly choosing CRC-32 as a data integrity algorithm.

WEP constructs a RC4 per-packet key by simply concatenating a known value, the initialization vector (IV), to a base key value. This construction allows the attacker to easily identify packets encrypted with weak keys as described by Scott Fluhrer, Itsik Mantin, and Adi Shamir [2] and thus facilitates recovery of the base key. In addition, the lack of replay protection, or the ability to repeatedly use the same IV values, coupled with the lack of a WEP key management protocol, facilitates the recovery of the base key. Once the base key has been compromised, the system is wholly vulnerable. Finally, the inappropriate choice of CRC-32 as a data integrity mechanism trivializes bit-flipping attacks. With such vulnerabilities, WEP is extremely susceptible to both passive and active attacks.

A task group within the IEEE 802.11 working group, TGi, is developing standards for improved wireless local area network (WLAN) security. TGi is also faced with the reality of millions of deployed IEEE 802.11b units. With this in mind, the TGi is adopting a short-term solution that will address WEP vulnerabilities in the deployed units as well as a long-term solution to fully address WLAN security. The short-term solution must be easily deployed without requiring customers to discard their hardware.

Nonetheless, both the short-term and the long-term solutions must provide the framework and elements critical to WLAN security. This paper describes such a framework.

2. Overview of Wireless LANs and WEP

The fundamental building block of the IEEE 802.11 WLAN architecture is the Basic Service Set (BSS). The BSS is a group of stations (wireless network nodes) located within a limited physical area, where each station is capable of communicating with

every other station. There are two WLAN design structures based on the BSS: infrastructure and ad hoc networks.

An infrastructure-based WLAN is composed of one or more BSS. Each station has exactly one BSS link to a connecting infrastructure, the Distribution System (DS), which allows access to external networks. The station's attachment point to the DS, called the Access Point (AP), relays packets from the other stations within the BSS to the DS as shown in Figure 1.

An ad hoc WLAN has no infrastructure, and therefore no ability to communicate with external networks. An ad hoc WLAN is normally created to permit multiple wireless stations to communicate with each other, requiring minimal hardware or management support. The BSS of an ad hoc WLAN is referred to as an independent BSS (IBSS).

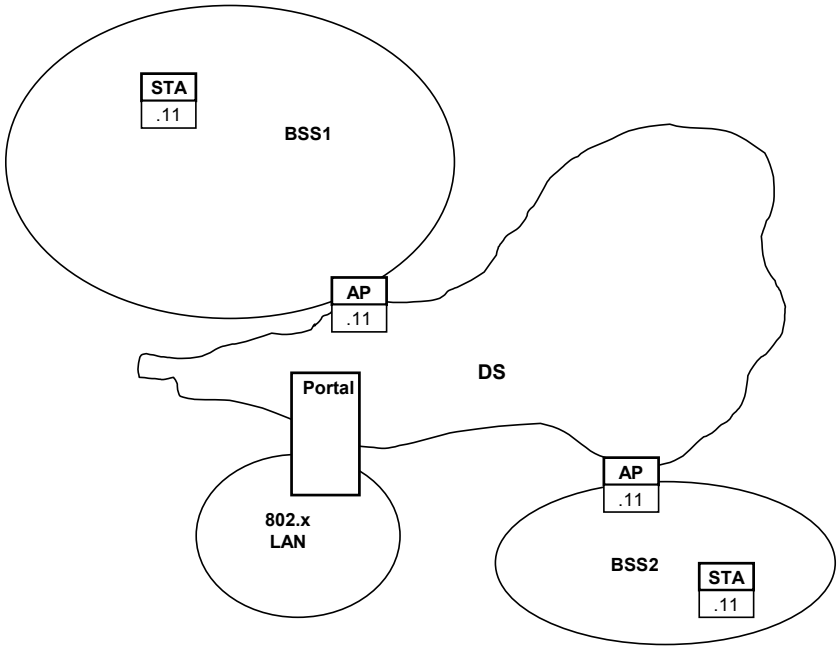

Figure 1. Typical Wireless LAN Configuration.

To transmit and receive, data stations compose packets called media access control (MAC) service data units (MSDUs). When transmitting data, the MAC layer determines whether the data in the MSDU should be partitioned into smaller fragments or MAC protocol data units (MPDU) that are then processed for transmission. Conversely, when receiving data, the MAC layer determines whether the MPDU is a fragment thus requiring reassembly of an MSDU. Each MPDU includes a *frame check sequence* (FCS); a CRC-32 is computed over the entire MPDU. The MAC uses the FCS to ensure that the frame has not suffered perturbations due to radio signals.

Prior to communicating data, stations and APs must establish and validate access to the network as well as establish whether to communicate openly (open authentication) or

securely (shared authentication). Open authentication is used to pass packets freely between the station and the AP, while shared authentication is used to protect packets. Open authentication is really not an authenticator; it allows any requesting stations to authenticate and enter the BSS. Shared authentication uses a challenge and response exchange along with a shared secret to authenticate the station to the AP, but is easily compromised. TGi plans to replace the flawed shared key authentication with authentication mechanisms running over IEEE 802.1X, but we will not discuss this effort further in this paper.

The IEEE 802.11 standard does not specify a means for obtaining the shared secret. The shared secret is typically a 40-bit key or a 104-bit key that is shared between many stations. A key that is shared between the AP and many stations is referred to as a default key. A key that is shared between the AP and only one other station is referred to as a key-mapping key. Both default keys and key-mapping keys are subsequently used to protect communications between associated stations.

The WEP protocol is used to protect MPDUs, the IEEE 802.11 packet fragments. WEP uses the pre-established shared secret key and the RC4 algorithm for encryption, and it uses CRC-32 to compute an *Integrity Check Value* (ICV). The ICV is computed over the MPDU data. The resulting 32-bit ICV is appended to the MPDU prior to encryption. The RC4 key is composed of a 24-bit IV value concatenated with the shared secret key to form a per-packet key. The MPDU data and ICV are then encrypted under the per-packet key. The IV and a key identifier are prepended to the encrypted MPDU data field, and the resulting WEP Protocol Data Unit ,shown in Figure 2, is ready for transmission to the peer.

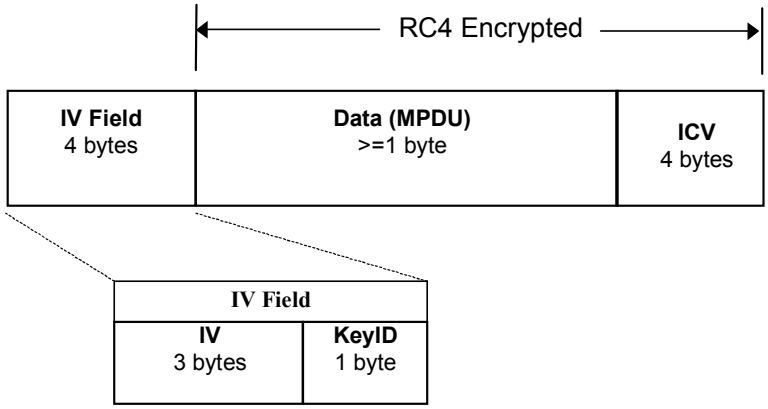

Figure 2. WEP Protocol Data Unit.

3. Review of WEP Flaws

This section reviews the major problems with WEP. This summarizes the work reported in [1], [2], [3], and [4]. The WEP design exhibits both sins of commission and sins of omission.

Foremost among the sins of commission is the misuse of the RC4 stream cipher. RC4 is an excellent cipher, used widely in a range of modern security applications, largely because of the high degree of privacy it affords with a relatively low performance penalty. However, stream ciphers are particularly difficult to use properly in packet protocols. RC4 was a particularly poor choice in the WEP context, as will be described later. To understand this, it is useful to review how stream ciphers operate.

By definition, a stream cipher generates a pseudo-random stream, called a *key stream*. The encryptor exclusive-ORs (XORs) the generated key stream with the plaintext to produce ciphertext. The decryptor generates the same key stream and XORs it with the ciphertext to recover the plaintext.

What happens if the encryptor XORs the same key stream with two different plaintexts? Compromise. Suppose two plaintext byte sequences p_1, p_2, p_3 and q_1, q_2, q_3 are both encrypted with key stream k_1, k_2, k_3. The corresponding ciphertexts are:

$$p_1 \oplus k_1, p_2 \oplus k_2, p_3 \oplus k_3 \qquad \text{and} \qquad q_1 \oplus k_1, q_2 \oplus k_2, q_3 \oplus k_3$$

If both of these two ciphertext streams are exposed to an attacker, then a catastrophic failure of privacy results, since:

$$(p_i \oplus k_i) \oplus (q_i \oplus k_i) = p_i \oplus q_i$$

That is, using a stream cipher to encrypt two plaintexts under the same key stream trivially leaks a great deal of information about plaintext. All stream ciphers come with a warning to never do this.

A second problematic stream cipher characteristic is the encryptor and decryptor must remain synchronized. That is, the decryptor must know the relative offset of each byte of ciphertext, otherwise it will use the wrong byte of key stream for decryption. Since the IEEE 802.11 MAC is neither reliable nor does it provide in-order delivery at the level at which WEP operates, a cipher with the *random access property* is needed. A cipher with the random access property can easily generate any specified byte of the key stream. This property is useful over an unreliable communications channel, because context accompanying a packet can specify the synchronization to the decryptor. RC4 does not have the random access property, but the WEP key derivation function (concatenation of the IV and the base key) simulates the random access property. Unfortunately, the RC4 key schedule is too lightweight to be used in this manner, and the resulting key streams have too many similarities.

The WEP designers had some intimation of these stream cipher limitations, and they tried to compensate for them. In order to avoid these problems they defined a per-packet RC4 key. This is a reasonable strategy, but the design introduces more problems in the way it implements the strategy.

The first per-packet key problem is the manner in which WEP constructs the per-packet key. The IV is concatenated with a base key. The encryptor selects the IV and transmits it as plaintext as described above. The decryptor uses the received IV to construct the same per-packet key. This construction should have been suspect at the outset, as it exposes the first part of the encryption key. In a paper presented in August 2001, Scott Fluhrer, Itsik Mantin, and Adi Shamir [2] investigated the simplistic RC4 key schedule when a portion of the RC4 key is known, and they showed that this kind of construction leads to a class of RC4 *weak keys*. Patterns in the keys themselves are reflected in patterns at the beginning of the generated key stream. If the first two bytes of enough key streams can be observed, then the whole RC4 key can be recovered, including the base key used to construct the per-packet keys for other packets. This exploit is called an FMS attack.

Using such a weak method to construct per-packet keys would be a problem in any case, but the WEP design compounds this by an unrelated situation: the first few bytes of encrypted data in almost every packet are known. The SNAP-SAP [12] header is almost always at the beginning of the packet. This allows an adversary to recover the first two bytes of the generated key stream, which is precisely the information needed to mount the FMS attack and recover the base key. AirSnort, a public domain hacker tool, implements this attack. The first version of AirSnort appeared in August 2001, and it had to examine about 1 million packets to recover the base key. By January 2002, the AirSnort implementation had reportedly been improved to require only about 20,000 packets to recover the base key; this represents about 11 seconds of IEEE 802.11b traffic under normal conditions.

As bad as this is, the IV usage also allows an attacker to recover all the plaintext without ever learning the base key, however. WEP uses a 24-bit IV, which means there can be a maximum of $2^{24} \approx 16$ million per-packet keys associated with any base key. Thus, to avoid duplication, the base key must be replaced at least once every 2^{24} packets. Since an IEEE 802.11b channel can sustain an average of about 1800 data packets a second, the base key must be replaced at least every 2.5 hours.

The collision of the 24-bit IVs suggests some very simple, low-tech attacks. A patient eavesdropper can record all the WEP encrypted traffic, group recorded packets by IV, and XOR packets encrypted under the same IV to learn a significant amount about the data itself. It is often feasible to use pattern recognition techniques to disentangle the two XORed plaintext packets and, once this is accomplished, the generated key stream can be exposed. This permits the eavesdropper to directly decrypt all subsequent packets until the base key changes. A less patient attacker can arrange to send known plaintext, such as SPAM e-mail into the network, to directly recover the key stream.

A third problem is IV selection. The WEP specification imposes no rules on IV selection; it only recommends changing the IV "frequently" – an undefined term. As a result, vendors implement their own IV selection strategy. Some implementations operate with a fixed IV, employing the same RC4 key to encrypt every packet! Other vendors selected the IV at random. After n packets the probability of an IV collision

under this strategy is $P_n = 1/2^{24}$ when $n = 2$ packets, and $P_n = P_{n-1} + (n-1)(1-P_{n-1})/2^{24}$ for $2 < n < 2^{24}$; $P_n = 1$, of course, for $n \geq 2^{24}$. Therefore, after only 4823 packets there is a 50% chance of collision. Other vendors pursued a third strategy, using the IV space as a circular counter, always starting at zero upon boot. This strategy guarantees a collision after two stations transmit a single packet protected by the same base key.

A passive eavesdropper can exploit the problems discussed thus far, but an active attacker can do even more damage. WEP fails to provide effective data integrity. The WEP designers thought they had designed a data integrity mechanism, but the specified algorithm fails to provide the intended protection. There are three problems with this design.

The first problem is the data integrity mechanism itself. The WEP transmitter computes a CRC-32 over the data payload, appending the resulting ICV to the data, and then encrypts the ICV along with the data. The idea was the receiver could detect data modifications by decrypting the data and the ICV and then verifying that the decrypted ICV matches the data. However, this algorithm does not prevent undetected data modification. An attacker can record a valid packet, create a zero pad with the same length of the encrypted data, flip one or more bits, and compute the ICV of this bit-flipped zero pad. Then the attacker can create a forgery by XORing both the bit-flipped zero pad and ICV and the encrypted data in the recorded packet, including the recorded encrypted ICV. This works because the CRC-32 construction and XOR-based encryption commute. That is, the same value results, regardless of the order of the operations. Further, the CRC-32 is linear over combinations of data it protects. After decryption, the modified ICV in the forgery will validate correctly, and WEP will accept the packet as genuine. If combined with packet analysis, the attacker can use this technique to construct packets with correct application data.

Even if the encrypted CRC-32 construction provided the intended protection, the data integrity mechanism does not cover all the information that needs to be protected from modification. As an example, the ICV mechanism does not protect the packet destination address. An attacker can record a packet from a station to an AP, change the destination address, and then send the packet. When the AP receives this forgery, it dutifully decrypts the packet and forwards it to the "wrong" address, perhaps to the attacker. A similar alteration of the source address of packets from the AP to another station allows the adversary to masquerade as any station.

The last data integrity mechanism problem: WEP provides no replay protection. An attacker can record any packet and then retransmit them later with or without alteration. Since each of the packets is encrypted under a valid key, they will be accepted at the IEEE 802.11 level as valid. Traffic analysis can reveal the use of various connectionless protocols, with the replayed data being accepted as authentic at the application layer.

All of these problems arise when an eavesdropper can collect a sufficient number of packets encrypted under the same base key. If the WEP base key were changed sufficiently often, then these attacks might afford an adversary significantly fewer options

to compromise security. However, IEEE 802.11 provides no mechanism to replace keys, practically requiring customers to use static, manually configured keys. It is infeasible to manually change keys often enough to provide protection from these attacks.

The WEP architecture compounds this problem in two ways. First, WEP uses the same key to protect data in both directions over a link. Implementations often use a counter in each direction to generate the next IV, which guarantees immediate IV collision and data exposure. Second, IEEE 802.11 only provides a way to name default keys, thereby encouraging the use of a single group key within a WLAN. It is simply infeasible to manage quantities that cannot be named.

4. Solution Constraints

Millions of WEP-based equipment have shipped and been deployed. The industry has an obligation to fix the security of this installed base if at all possible. Like most modern communication equipment, IEEE 802.11 devices are comprised of hardware and software. WLAN hardware has been designed as a commodity, so it is not cost effective to add or swap out particular hardware chips in a WLAN device; instead, it is cheaper to replace the entire hardware unit. This implies that WEP patches operating on already-deployed IEEE 802.11 hardware will rely entirely on software upgrade. This is the first design constraint, and it poses a particularly sticky dilemma.

IEEE 802.11 APs present a computational bottleneck, as they have little spare processing capacity. Recall that in an infrastructure deployment, all stations link with the AP instead communicating directly among themselves, and the AP handles every message exchanged within the BSS. In order to be competitive in a commodity market, APs are typically implemented with the cheapest hardware possible, using a microprocessor like an i486, ARM7, or PowerPC running at 40 or even 25 MHz. The load generated by normal WLAN traffic often consumes 90% or more of the microprocessor computational bandwidth, so very few cycles are available for new functions. In some cases, there may only be 2 million unused instructions per second available. This is the second major design constraint.

This is an impassible barrier for a traditional security design targeted for implementation in software. The cryptographic functions such a design would necessarily employ are processor intensive. At IEEE 802.11b data rates, standard cryptographic primitives can easily consume the entire AP processor.

Since they support WEP, how do CPU constrained devices implement RC4 encryption? Nearly all shipping APs have custom hardware to handle the RC4 encryption. Most of this hardware is tuned to construct per-packet keys according to the WEP algorithm: the per-packet key is a base key concatenated to an IV, which appears as plaintext in each packet. On transmit, the hardware expects the packet as it input, along with the base key and IV. The custom hardware constructs the per-packet key, encrypts the MPDU payload, inserts the IV, and passes result to the radio transmitter. On receive, the

hardware extracts the IV, locates the base key, constructs the per-packet key, and decrypts the MPDU payload as it arrives from the radio receiver. In most receivers there is very little time between packet arrival and start of decryption. In this time interval, at most three hundred instructions can be executed. In some devices, some of this time is used to locate the base key. The hardwired encryption function represents a third major design constraint. The design affords few opportunities for software intervention into an outgoing packet after encryption and even fewer for an arriving packet prior to decryption.

On first analysis, therefore, fixing WEP with any cryptographically sound approach seems to be impossible without instantly obsolescing all existing hardware. TGi is designing a long-term solution that does precisely this. It is impossible to utilize standard cryptographic functions in any way to rescue WEP, at least on already-deployed hardware, because very few have sufficient spare processing capacity to accommodate the needed operations.

The alternative within present hardware is to do nothing. However, TGi is designing a short-term solution with vastly improved security; however, the cost, performance, and security trade-offs required to support deployed hardware does not allows these WEP repairs to fully address the TGi security goals. The WEP repairs presented in this paper serve as a short-term solution to allow security improvements on currently deployed hardware until the long-term solution becomes available.

5. WEP Repairs

Four components comprise the WEP security repairs. Two components, rapid rekeying and an improved per-packets key derivation function, provide protection against passive attacks. The other two components, data integrity checking and replay prevention, provide protection against active attacks. All four components are necessary for a complete security solution.

The rapid rekey component of the short-term and long-term solutions is still under development. While TGi continues to make rapid progress in this area, it is less stable than the other three components of the solution. In the next section, we discuss the evolving rapid rekey component, and we provide insight to the various elements that will likely be included. Following the rapid rekey discussion, we discuss the more stable components: the per-packets key derivation function, data integrity checking, and replay prevention.

5.1. Rapid Rekeying

A key must be refreshed when its lifespan has expired or when an attack is presumed. The lifespan of a particular key depends on the encryption algorithm and the way that the key is used. In the WEP protocol, the use of a 24-bit IV suggests a key lifespan of $2^{24} \approx$

16M packets. However, because of the security flaws discussed above, some experts suggest that the maximum key lifespan for the WEP base key ought to be no more than about $2^8 \approx 256$ packets. Using IEEE 802.11b average packet and data rates, a key refresh every 256 packets would demand a new key every 0.2 seconds, a prohibitive rate. The short-term solution (using the per-packet key derivation function discussed in the next section) employs a 16-bit IV, allowing the key to survive up to $2^{16} \approx 64$K packets, and the key lifespan is about 1 minute. The long-term solution uses AES [11] with at least a 28-bit IV, allowing the key to survive up to $2^{28} \approx 268$M packets between rekeys, and the key lifespan is about 19 hours. While all solutions require key refreshment, the rekey demands of WEP and the short-term solution warrant a framework that allows for rapid rekeying.

Protocols for key management and key refreshment are well known and practiced today, typically in layers above the MAC. Mechanisms such as IKE [7] and TLS-EAP [8] facilitate the establishment of secret keys. While these protocols are well suited for their intended use, they lack some of the characteristics necessary to establish MAC layer keys while minimizing disruption to the traffic flow. To overcome this shortcoming, a new protocol based on the IEEE 802.1X [9] framework is being defined.

The solution includes a three tier key hierarchy. First there is a *master key*, which is established at initial contact between a station and a BSS. The master key is used subsequently to derive fresh *EAPOL keys*. The architecture uses the EAPOL (EAP over LAN) keys to protect data distributed by IEEE 802.1X EAPOL key messages. Finally the EAPOL key messages disseminate information used to derive the *temporal key*s (TK), which are used with the encryption and data integrity algorithms.

Each of these key types has freshness requirements. The station and the AP must establish a new master key each time the station comes into contact with the BSS after being away long enough for its master key to lapse. The protocol allows the station and the AP to establish fresh EAPOL keys on each reassociation. The protocol also allows the establishment of a fresh TK before the IV space is exhausted. In general, the establishment will begin when either the AP or station has consumed all but the last 1000 IV values.

Before the IEEE 802.1X EAP framework can be used to establish a master key, a cooperative relationship, or *secure session*, must be established between the communicating stations. The *secure session* establishment makes use of cryptographic keying material and attributes shared by the AP and the station to securely exchange and synchronize encryption keys. Without cooperative establishment of these attributes, the AP and station cannot synchronize key state and authenticate rekey events. Updates are being made to IEEE 802.11 and IEEE 802.1X to accomplish secure session establishment and are outside the scope of this paper.

The IEEE 802.1X EAPOL Key message supports the establishment of cryptographic keying material, which is then used to derive the TK, which includes keying material for encryption and data integrity. The EAPOL keys are used to authenticate the rekey event,

and to protect the exchanged keying material. TK derivation will likely include a nonce from each party, a counter, and the cryptographic keying material. Characteristics such as replay protection and key identification are also needed to ensure that the key refresh events are secure and synchronized. These characteristics are absent or deficient in the IEEE 802.1X protocol, and thus the IEEE 802.1X EAP framework and EAPOL Key message are being modified to meet these goals. AKEP, a protocol published by Bellare and Rogaway [10], exemplifies a key exchange that does meet the security goals. The AKEP protocol presents an authenticated three-way handshake and required attributes to securely establish a key. However, given the high rekey frequency demanded by the short-term solution, a three-way handshake is considered too costly, thus the single EAPOL Key message is being investigated and modified to provide the required security and synchronization.

Default keys can be shared between an AP and more than one station while key-mapped keys are unique to an AP and a single station. Naturally, a default key and a key-mapped key need to be refreshed under different conditions. The conditions necessary to refresh a key-mapped key are straightforward. Both the AP and the station must:

- Share the cryptographic material necessary to authenticate the key refresh event;
- Share the cryptographic material to either derive the new key or validate it, depending on whether a key distribution or key exchange is used;
- Generate a replay protection value to thwart adversaries from desynchronizing and spoofing the link; and
- Assign the same key identifier to the resulting key, allowing the AP and station to agree when the new key will be used (and the old key will be discarded).

Either a key distribution mechanism or key exchange handshake is suitable to refresh a key-mapping key used to protect the link between one station and the AP.

When refreshing a default key, the rekey protocol must allow multiple stations to securely switch to the new key at the same time, while adhering to the single link rekey conditions. A distribution mechanism using either a single message (IEEE 802.1X EAPOL Key message) or even the AKEP three-way handshake presents synchronization issues. If messages are used to deliver the new key or instruct the derivation of a fresh key, the WLAN can only deliver such messages serially. Serial message delivery presents a potential security breach unless the messages are guaranteed to arrive and be processed prior to key lifespan expiration. If a key is being refreshed due to the detection of an attack, serial delivery of key refresh messages to each station becomes prohibitive. A natural key distribution mechanism is to introduce the required attributes to authenticate and synchronize a new key into the beacon element. However, in an endeavor to maintain the same rekey protocol for both key-mapping and default keys, the IEEE 802.1X EAP framework is being investigated and enhanced to allow for a secure synchronization of keys. If instead of a broadcast mechanism, a serial message mechanism is used, a window of time must be well defined to allow for all stations to securely install the new key without disrupting MAC layer communications.

5.2. Per-packet Key Derivation

As previously described, WEP generates a different RC4 key for each packet by concatenating the 24-bit IV and the 104-bit (or 40-bit) base key. Using this method, the keys for different packets are too similar. As readily demonstrated by the FSM attack, the lightweight RC4 key-scheduling algorithm is vulnerable when used this way, particularly when the initial few bytes of plaintext are easily predictable, as is almost always the case with IEEE 802.11. A short-term solution must therefore introduce an improved key derivation function.

Note that the long-term solution will use AES, not RC4, for encryption. AES has significantly different properties that obviate the security requirement for per-packet keys, so there is no need for per-packet key derivation function in the long-term solution.

Ron Rivest, the author of RC4, suggests two solutions to the weaknesses in the RC4 key-scheduling algorithm. He recommends discarding the first 256 output bytes of the key stream, or he recommends strengthening the key-scheduling algorithm by preprocessing the key and the IV by passing them through a one-way hash function such as MD5 [13]. However, discarding the first 256 output bytes is too expensive for existing equipment, and it is infeasible for some implementations. One-way hash functions, such as SHA-1 [14] and MD5, are also too expensive for already deployed equipment.

Since the obvious fixes are too expensive, a new key derivation function was designed that is cheap enough to execute on existing hardware. It derives a per-packet key from the 128-bit TK. This solution will likely be distributed as a firmware upgrade by vendors, allowing their customers to update existing vulnerable equipment.

The new per-packet key derivation function operates in two phases. In the first phase, the transmitter address (TA) is mixed into the TK. By including the TA, multiple stations can use the same TK, and each station (including the AP) will generate a different key stream. This property is important in all networks. Consider the simple case where a station communicates only with one AP. Data sent by the station to the AP and data sent by the AP to the station will be encrypted with the same TK. If the TA were not mixed with the TK, the same series of RC4 key streams would be used by both the station and the AP, enabling data recovery attacks discussed above.

The output of the first phase will likely be cached; it can be reused to process subsequent packets that use the same TK and the same TA.

The second phase uses a Feistel cipher to mix the IV and the output of the first phase. The IV is a 16-bit counter, initialized to zero when the TK is established. By including the IV, each packet will be encrypted with a unique key stream. Using a Feistel cipher to perform the mixing makes it difficult for an attacker to correlate the IV and the per-packet key. The two-phase per-packed key derivation process is summarized in Figure3.

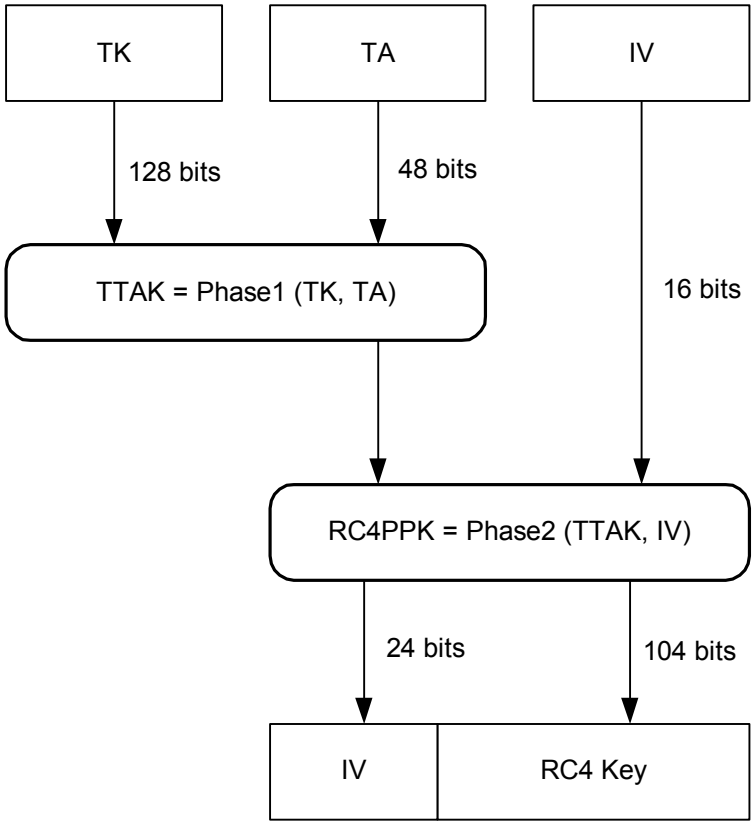

Figure 3. Per-packet RC4 Key Derivation.

Since the key derivation function must easily integrate with existing hardware, the outputs are a 24-bit IV and a 104-bit RC4 key. These are concatenated to form the RC4 per-packet key, just like WEP.

5.3. Data Integrity

WEP fails to prevent packet forgery. The addition of a *Message Integrity Code* (MIC) attempts to correct this deficiency by ensuring that the data within the packet and selected portions of the packet header have not been modified. That is, the data sent by the originator and the data received by the recipient are the same. The MIC is a keyed cryptographic function, traditionally called a Message Authentication Code (MAC). Using the term "MIC" avoids confusion with Medium Access Control, with the same acronym.

The MIC value is computed from the key, the data within the packet and selected portions of the packet header, and then the MIC value is appended to the packet data. When the packet data is encrypted, the MIC value is also encrypted.

The near-term solution must accommodate the very limited amount of processing power available in the MAC processor (as well as some APs), so a relatively weak integrity function is employed. This is acceptable because:

1. The ICV (a CRC-32 checksum) is computed on the plaintext data payload. The ICV is part of WEP, and most implementations compute it in hardware.
2. The FCS (another CRC-32 checksum) is computed on the ciphertext. The FCS is used to detect transmission errors.
3. The MIC value is encrypted, which indirectly increases the overall strength.

To accommodate implementation of the MIC algorithm in the host processor or the MAC processor, which ever has the available processing capacity; the MIC is computed on the MSDU (before fragmentation into MPDUs). The host does not generally have access to MPDUs, and the host is completely unaware of any fragmentation or reassembly.

A new MIC algorithm, called *Michael* [16], was developed for the short-term solution. Michael was designed to be efficient on currently deployed MAC processors as well as processors typically used in APs.

Performance and security are the two dominating concerns in Michael's design. Its inner loop uses only XORs, shifts, byte-swaps, and additions; all of these operations are cheap on the target processors. Michael costs about 3.5 cycles/byte on an ARM7, and about 5.5 cycles/byte on an i486. This means it will consume about 3.1 M cycles/second on an ARM7-based AP, and 4.8 M cycles/second on an i486-based AP. This processing ought to consume every unused processor cycle on many first generation IEEE 802.11 APs, resulting in some performance degradation in a fully loaded BSS.

It is easy to establish an absolute upper bound on the security afforded by any MIC. When an n-bit MIC algorithm is employed, the algorithm maps any message to one of 2^n possible values. As a result, MIC security level is usually measured in bits. If the security of a MIC is s bits, then by definition the probability an attacker can construct an acceptable forgery on the first packet is 2^{-s}, and by the birthday paradox, an adversary expects to produce an acceptable forgery after about $2^{s/2}$ packets. If the MIC algorithm is completely secure, then the number of bits in the MIC value is the number of bits of security provided. However, Michael, by design, sacrifices security for computational efficiency. Even though Michael has a 64-bit MIC value, it was designed with a target security level of 20 bits, and it is believed that it slightly exceeds this target. The best attack known against Michael is based on differential cryptanalysis, and it indicates that Michael offers 29 bits of security.

Accidental MIC check failures will occur very rarely. The FCS will detect noise or interference on the radio channel. A station receiving 100 randomly formatted packets per second can expect one to pass the FCS checks less than once a year. As discussed above, the FCS and the ICV CRC-32 checksum computations use the same polynomial. Therefore, if a modified packet passes the FCS check, it will most likely also pass the ICV check. However, the encryption between the FCS and the ICV do ensure that the

receiver and the originator are using the same TK. Only packets that satisfy both the FCS and ICV checks will get to the point where the MIC is verified.

Given the very low rate of accidental MIC failures, it is reasonable to assume that an active attack in progress. Countermeasures to thwart the active attacker are deployed. These countermeasures include discarding the current TK, closing the association, and notifying the system administrator. Discarding the current TK prevents the attacker from learning anything about the TK from the MIC failure. Shut down of the association introduces delay, and this slow down prevents the attacker from sending a large number of fraudulent packets in a short time. Notification of the system administrator allows a human to detect the location of the active attacker's transmitter.

In the long-term solution, an AES-based data integrity mechanism will be used to detect MPDU modification. By protecting the MPDU, AES modes that provide both confidentiality and data integrity can be employed in the long-term solution. Such modes provide protection against modification for the encrypted data and for plaintext MPDU header fields. Also, by providing both services on the MPDU, replay detection can take advantage of an integrity protected sequence number. Replay detection is discussed further in the next section.

5.4. Replay Detection

In order to provide replay protection, the short-term solution uses the existing WEP IV field as a packet sequence number. Each TK has its own sequence number space.

The transmitter initializes the sequence number to zero whenever a new TK is set, and then increments the sequence number for each successive MPDU. If the TK is not refreshed prior to IV sequence space exhaustion, the transmitter must halt communication.

The receiver follows the same initialization rule, resetting a sequence number to zero when the TK is refreshed. A packet is considered to be out of order when its sequence number is the same or smaller than a previously received MPDU associated with the same TK. If a MPDU arrives out of order, then it is considered to be a replay; it is discarded, and a MIB counter is incremented. The receiver increments the replay counter only if the ICV of the MPDU is valid and the sequence number indicates in-order delivery. As described in the previous section, the long-term solution will provide data integrity protection for the MPDU header fields. This will provide modification detection for the IV field, which is used as the sequence number.

In the short-term solution, the data used by replay detection algorithm is protected indirectly. Since the IV is used to construct a per-packet key, modification of the IV will cause the receiver to attempt decryption with the wrong key stream. Thus, the packet data and the ICV will decrypt incorrectly, leading to an ICV verification failure.

The replay detection algorithm relies on the fact that IEEE 802.11 preserves the packet sequence. However, IEEE 802.11 TGe is working on a *quality of service* (QoS) definition that obviates this assumption. Once the TGe QoS work is deployed, three options will be available:

- QoS and security cannot be used together.
- Extend key management to assign a different TK for each QoS traffic class, each with its own sequence number space.
- Extend the replay detection mechanism to include a per-traffic-class replay counter value.

As of this writing, IEEE 802.11 TGi has not yet decided which approach it will employ, although the second alternative is likely.

5.5. Interdependence

A WEP security patch requires all of these enhancements. If rapid rekeying is not implemented, then the resulting protocol is still subject to data compromise when IVs collide. If the per-packet key derivation algorithm is not implemented, then the protocol is still subject to the FMS attack. If the replay detection algorithm is not implemented, then the protocol is still subject to forgeries by replay. And if the message integrity check is not implemented, the protocol is still subject to packet forgery attacks. Trying to implement only some of these enhancements would be similar to closing only some of the hatches on a submarine before submerging: doing so fails to achieve the ultimate goal. Security becomes possible only when all the core deficiencies are addressed.

The replay detection and MIC together defend against active attacks. However counter-intuitive it might be, it is always necessary to try to defeat active attacks that undermine data integrity to achieve confidentiality guarantees, because they can be turned into attacks against the encryption itself, as they cause the protocol itself to reveal more about the encryption key than passive attacks.

Rapid rekeying and the improved key derivation function together restore the assumptions made by the encryption algorithm. Without these guarantees, the encryption function cannot do its job properly.

6. Interoperability Considerations

Interoperability must be preserved during the transition from WEP to the short-term solution to the long-term solution. To accomplish this, each station must know the protocol and algorithms that are being used by its peers.

In a BSS, the AP is the only peer. At association establishment, the station offers its preferences for authentication algorithms and the cipher suite for the protection of unicast and multicast traffic. If none of the alternatives offered are acceptable, then the AP must

reject the association; otherwise the AP selects one of the authentication algorithms, one of the unicast cipher suites, and one of the multicast cipher suites. An EAP mechanism over IEEE 802.1X is the only non-proprietary authentication algorithm specified for use with the short-term solution. EAP-TLS [8] and an EAP mechanism based on SRP [15] would be suitable. Both EAP mechanisms provide mutual authentication and key agreement. EAP Authentication mechanisms without these properties should not be used, as they fail to provide fresh master keys. Further, use of the flawed legacy IEEE 802.11 authentication is prohibited with the short-term and long-term solutions. IEEE 802.11 TGi has not yet defined a similar protocol for ad hoc networks. It remains an open work item. However, given the small number of stations that are generally members of an ad hoc network, the unicast cipher suite and the multicast cipher suite could be manually selected.

7. Conclusion

IEEE 802.11 TGi has defined an architecture that repairs the known deficiencies in WEP. Four components comprise the WEP security repairs. Rapid rekeying and an improved per-packet key derivation function provide protection against passive attacks. Message integrity checking and replay prevention provide protection against active attacks. All four components are needed for a complete security solution.

IEEE 802.11 TGi is developing a short-term solution to address the WEP vulnerabilities on currently deployed hardware. Though the short-term repairs do not provide the security strength as that of the long-term solution, it allows currently deployed hardware to persist until the new long-term solution becomes available.

Simultaneously, IEEE 802.11 TGi is developing a long-term solution to fully address WLAN security needs, adopting the AES algorithm.

While this work is still in progress, it is clear that solutions will be available soon.

References

[1] Borisov, N., I. Goldberg, and D. Wagner, "Intercepting mobile communications: the insecurity of 802.11," in *Proc. International Conference on Mobile Computing and Networking*, ACM, July 2001, pp 180-189.

[2] Fluhrer, S., I. Mantin, and A. Shamir, "Weaknesses in the key schedule algorithm of RC4," in *Proc. 4th Annual Workshop on Selected Areas of Cryptography, 2001*.

[3] Stubblefield, A., J. Ioannidis, and D. Rubin, "Using the Fluhrer, Mantin, and Shamir attack to break WEP", AT&T Labs Technical Report TD-4ZCPZZ, AT&T Labs, August 2001.

[4] Walker, J., "Unsafe at any key size: an analysis of the WEP encapsulation," IEEE 802.11 doc 00-362, October 27, 2000.

[5] Shirey, R., "Internet Security Glossary," RFC 2828, May 2000.

[6] IEEE Std 802.11, Standards for Local and Metropolitan Area Networks: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[7] Harkins, D., and D. Carrel, "The Internet Key Exchange (IKE)," RFC 2409, November 1998.

[8] Aboba, B., and D. Simon, "PPP EAP TLS Authentication Protocol," RFC 2716, October 1999.

[9] IEEE Std 802.1X, Standards for Local and Metropolitan Area Networks: Standard for Port based Network Access Control, 2001.

[10] Bellare, M., and P. Rogaway, "Entity Authentication and Key Distribution", Crypto '93 Proceedings, August 1993.

[11] National Institute of Standards and Technology. FIPS Pub 197: Advanced Encryption Standard (AES). 26 November 2001.

[12] Postel, J., and J.K. Reynolds, "Standard for the transmission of IP datagrams over IEEE 802 networks," RFC 1042, February 1988.

[13] Rivest, R., "The MD5 Message-Digest Algorithm," RFC 1321, April 1992.

[14] National Institute of Standards and Technology. FIPS Pub 180-1: Secure Hash Standard. 17 April 1995.

[15] Wu, T., "The Secure Remote Password Protocol", In Proceedings of the 1998 *Internet Society Symposium on Network and Distributed Systems Security*, San Diego, CA, pp. 97-111.

[16] Ferguson, N., "Michael: an improved MIC for 802.11 WEP," IEEE 802.11 doc 02-020, January 17, 2002.